EXHIBIT 3

GOOGLE INC.'S REQUEST FOR JUDICIAL NOTICE AND FOR CONSIDERATION OF DOCUMENTS REFERENCED AND RELIED UPON IN THE CONSOLIDATED AMENDED COMPLAINT

WEB POLICY

a blog about technology, policy, and law
by Jonathan Mayer, a grad student at Stanford

FEB 17 2012

DO NOT TRACK, MEASUREMENT, PRIVACY

Safari Trackers

Apple's Safari web browser is configured to block third-party cookies by default. We identified four advertising companies that unexpectedly place trackable cookies in Safari. Google and Vibrant Media intentionally circumvent Safari's privacy feature. Media Innovation Group and PointRoll serve scripts that appear to be derived from circumvention example code.

In the interest of clearly establishing facts on the ground, this post provides technical analysis of Safari's cookie blocking feature and the four companies' practices. It does not address policy or legal issues. (More on that soon.)

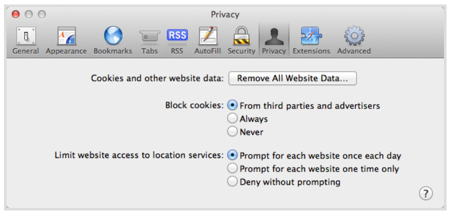
Before proceeding further, I want to thank the countless friends and colleagues who provided invaluable feedback on this project. In particular: ★★★★★, whose insights have been vital at every step, and Ashkan Soltani, whose crawling data was instrumental in uncovering PointRoll's practices and understanding the prevalence of cookie blocking circumvention.

Third-Party Cookie Blocking in Safari

Every popular web browser, save Opera Mini and the Android built-in browser, includes a "third-party cookie blocking" privacy feature. (The remainder of this post uses the term "cookie blocking" for brevity.) These options share a common high-level purpose: impose limits on cookies from "third-party domains," that is, domains that differ from the "first-party domain" in the browser's URL bar. In practice, however, implementations vary substantially; for (slightly out-of-date) specifics, see the Center for Democracy and Technology's 2010 Browser Privacy Features report and Google's Browser Security Handbook.

Safari's cookie blocking feature is unique in two ways: its default and its substantive policy.

Unlike every other browser vendor, Apple enables cookie blocking by default. Every iPhone, iPad, iPod Touch, and Mac ships with the privacy feature turned on.

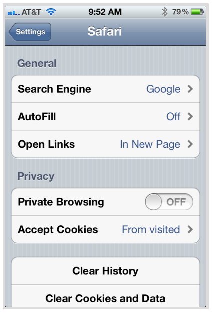


Cookies and other website data: Remove All Website Data...

Block cookies: [✓ From third parties and advertisers] Always Never

Limit website access to location services: [✓ Prompt for each website once each day] Prompt for each website one time only Deny without prompting

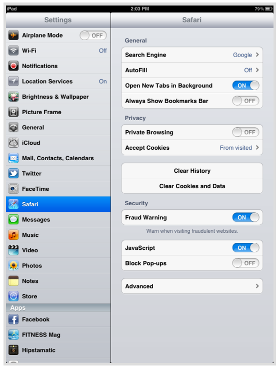
Default Privacy Settings in Desktop Safari



General
Search Engine: Google
AutoFill: Off
Open Links: In New Page

Privacy
Private Browsing: OFF
Accept Cookies: From visited

Clear History
Clear Cookies and Data



General
Search Engine: Google
AutoFill: Off
Open New Tabs in Background: ON
Always Show Bookmarks Bar: OFF

Privacy
Private Browsing: OFF
Accept Cookies: From visited

Clear History
Clear Cookies and Data

Security
Fraud Warning: ON
Warn when visiting fraudulent websites.
JavaScript: ON
Block Pop-ups: OFF
Advanced

Default Privacy Settings in iPhone and iPad Safari

Apple advertises cookie blocking by default as a benefit of choosing Safari.

> Some companies track the cookies generated by the websites you visit, so they can gather and sell information about your web activity. Safari is the first browser that blocks these tracking cookies by default, better protecting your privacy. Safari accepts cookies only from the current domain.

Apple's cookie blocking policy is less restrictive than many competing browser vendors'.[1]

- **Reading Cookies** Safari allows third-party domains to read cookies.
- **Modifying Cookies** If an HTTP request to a third-party domain includes a cookie, Safari allows the response to write cookies.
- **Form Submission** If an HTTP request to a third-party domain is caused by the submission of an HTML `form`, Safari allows the response to write cookies. This component of the policy was removed from WebKit, the open source browser behind Safari, seven months ago by Google engineers. Their rationale is not public; the bug is marked as a security problem. The change has not yet landed in Safari.

These allowances in the Safari cookie blocking policy enable three potentially undesirable behaviors by advertising networks, analytics services, social widgets, and other "third-party websites."

- If a company operates both a first-party website and a third-party website from the same domain, visitors to the first-party website will be open to cookie-based tracking by the third-party service. Yahoo! is an example: it hosts both first-party websites and third-party advertising services on the `yahoo.com` domain.
- If a third-party website's content ever manages to load in a full browser window, the website can set cookies on its domain. An advertising company, for example, could set tracking cookies on its domain with a pop-up, pop-under, or temporary redirect (e.g. when a user clicks an ad).
- A third-party website can use JavaScript to submit a `form` in an `iframe` without user interaction.

This post focuses on the last behavior. We discovered four advertising companies that surreptitiously submit a `form` in an invisible `iframe` and place trackable cookies in Safari: Google, Vibrant Media, Media Innovation Group, and PointRoll. The balance of the post details each company's business practices.

Google

Google has, historically, operated most of its first-party websites on the `google.com` domain and most of its third-party services on other domains. For example: Google Analytics is served from `google-analytics.com`, Google software libraries are hosted at `googleapis.com`, Google static content is at `gstatic.com`, and Google's advertising services are on `doubleclick.net`.

Separating first-party websites from third-party services improves security: interactions between `google.com` content and other websites could introduce vulnerabilities. The domain separation also benefits user privacy: Google associates user account information with `google.com` cookies. By serving its third-party services from other domains, Google ensures it will not receive `google.com` cookies, and therefore will not be able to trivially identify user activities on other websites.

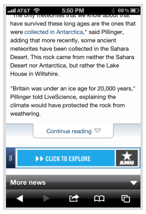
But what about when Google *does* want to identify the user on a non-Google website? Social personalization requires[2] just that! Google has two design options.

First, Google could embed `google.com` content on non-Google websites. This is the approach it has taken with its social sharing widgets; both the (defunct) Buzz button and the +1 button load resources from `google.com`.

Second, Google could synchronize information from its `google.com` domain to another domain, a process called "cookie syncing" in online advertising lingo. This is the approach Google took with `youtube.com` after it acquired YouTube. And this is the approach Google settled on for social personalization of `doubleclick.net` display advertising. Google announced the +1 button for display ads last September; here are the steps in the underlying cookie syncing technology, based on conversations with Googlers and an explanatory document that Google provided.

1. A display advertisement includes the cookie syncing mechanism's `iframe`, located at `http://googleads.g.doubleclick.net/pagead/drt/s`. We observed the `iframe` load in both desktop and mobile display ads. Here are example ads that included the

mechanism, from the Washington Post (Safari on Mac OS X) and MSNBC (Safari on iPhone).


"have survived these long ages are the ones that were collected in Antarctica," said Pillinger, adding that more recently, some ancient meteorites have been collected in the Sahara Desert. This rock came from neither the Sahara Desert nor Antarctica, but rather the Lake House in Wiltshire.

"Britain was under an ice age for 20,000 years," Pillinger told LiveScience, explaining the climate would have protected the rock from weathering.

Google Ads Including the `google.com` → `doubleclick.net` Cookie Syncing Mechanism

We also saw what appeared to be a special use of the mechanism on `youtube.com`, where Google placed an invisible advertisement in the footer that included the cookie syncing `iframe`.

In a FourthParty crawl of the homepages of the Alexa U.S. top 500 websites, we detected the cookie syncing mechanism on 39 pages. This figure likely underestimates the prevalence of Google's cookie syncing code since many websites show less advertising on their homepage. We observed the mechanism on New York Times and MSNBC article pages, for example, but not on their homepages.

2. The `iframe` loads a page that contains a `meta` refresh to http://google.com/pagead/drt/ui.

```
<!DOCTYPE HTML PUBLIC>

<html>

  <head>

    <meta http-equiv="refresh" content="0;url=http://google.com/pagead/d

  </head>

</html>
```

Apologies for any overflow; here and throughout the post I err on the side of preserving original formatting.

3. The response at http://google.com/pagead/drt/ui depends on whether the user is logged into Google. If the user is not logged in, the response includes a `Location` header that directs the browser back to `googleads.g.doubleclick.net` with some information in the `p` and `ut` parameters of the `Request-URI`.

```
Location: https://googleads.g.doubleclick.net/pagead/drt/si?p=CAA&ut=AFA
```

If the user is logged in, the response directs the user to Google's authentication service.

```
Location: https://accounts.google.com/ServiceLogin?service=doritos&passi
```

The authentication service then directs the user back to `googleads.g.doubleclick.net`.

(A quick explanation of the "Doritos" reference—as I understand it, that's Google's internal codename for social personalization of third-party display advertising.)

```
location:https://googleads.g.doubleclick.net/pagead/drt/si?p=CAEY9cLA-gQ
```

Google's documentation suggests that the `p` and `ut` parameters include an encryption of the user's login state and, if logged in, account ID. (The Google design document makes a number of technical claims about how the syncing mechanism preserves user privacy. This post does not address those claims.)

4. In a browser other than Safari, the response sets a "_drt_" social personalization cookie on `.doubleclick.net`. If the user is not logged into Google, the cookie's value is "NO_DATA", and the cookie is set to expire in 12 hours.

```
set-cookie:_drt_=NO_DATA; expires=Fri, 17-Feb-2012 13:37:41 GMT; path=/;
```

If the user is logged into Google, the "_drt_" cookie contains an encryption of the user's Google account ID, set to expire in 24 hours.

```
set-cookie:_drt_=AFkicjesF-jVECSOLRa1a-hf14FYVKIPEu4goDlxZZdVaxh1D4gDfJ6
```

My understanding is that the cookie expirations are set to limit syncing frequency. If the user was not logged in at last sync, a sync will not be attempted for at least 12 hours; if the user was logged in, a sync will not be attempted for at least 24 hours.

In early testing, we a noticed a "PREF" cookie was sometimes set at the same time as the "_drt_" cookie.

```
set-cookie:PREF=ID=5a7be344032983bc:TM=1325643281:LM=1325643281:S=-BWpqD
```

The behavior appeared to stop before we contacted Google about our findings. We have not received information from Google explaining the "PREF" cookie on `googleads.g.doubleclick.net`. It appears to have the same format as the "PREF" cookie on `google.com` and the same two-year expiration.

So far, a (relatively) straightforward cookie syncing mechanism. But we noticed a special response at the last step for Safari browsers. We tested 400 `User-Agent` strings to verify that this is a special case; a spreadsheet of results is available.

Instead of responding with the "_drt_" cookie, the server sends back a page that includes a `form` and JavaScript to submit the `form` (using POST) to its own URL.

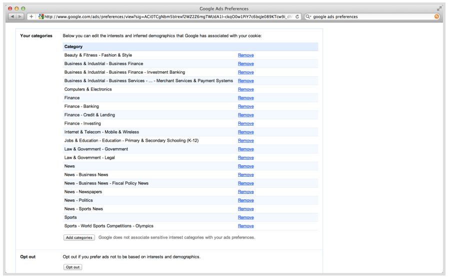
```
<html>

<head></head>

<body>

  <form id="drt_form" method=post action="/pagead/drt/si?p=CAA&amp;ut=AFAKxl(

  <script>
```

```
        document.getElementById('drt_form').submit();

      </script>

    </body>

    </html>
```

The response to the form submission then includes the `Set-Cookie` header for the "_drt_" cookie.

Recall that if a cookie is sent with an HTTP request, Safari's blocking policy will allow the response to write cookies. Owing to the "_drt_" cookie, all `doubleclick.net` content is now immunized from Safari's cookie blocking policy. The next time Google advertising content attempts to install the "id" tracking cookie for `.doubleclick.net`, it will successfully set. That next attempt may not even require that the user visit another page: We noticed that many Google ads periodically send requests to `doubleclick.net`, especially to a URL with the base `http://ad.doubleclick.net/activity`. A response to one of these requests can include a `Set-Cookie` header for the "test_cookie" cookie, which Google uses to make sure cookies successfully set (presumably to avoid wasting IDs and associated resources). A response to a subsequent request may then include a `Set-Cookie` header for the "id" cookie.

We confirmed that Google's `doubleclick.net` "id" cookie was functioning in Safari by observing behavioral interest categories appear in Google Ads Preferences. Here is an example set of inferred interests after browsing the New York Times website.


Google Ads Preferences
http://www.google.com/ads/preferences/view?sig=ACi0TCgNbm5blrexf2WZ2Z6mgTWUdA1I-ckqO0w1PiY7c6bqJe089KTcw9i_dM
google ads preferences

Your categories Below you can edit the interests and inferred demographics that Google has associated with your cookie:

Category	
Beauty & Fitness - Fashion & Style	Remove
Business & Industrial - Business Finance	Remove
Business & Industrial - Business Finance - Investment Banking	Remove
Business & Industrial - Business Services - ... - Merchant Services & Payment Systems	Remove
Computers & Electronics	Remove
Finance	Remove
Finance - Banking	Remove
Finance - Credit & Lending	Remove
Finance - Investing	Remove
Internet & Telecom - Mobile & Wireless	Remove
Jobs & Education - Education - Primary & Secondary Schooling (K-12)	Remove
Law & Government - Government	Remove
Law & Government - Legal	Remove
News	Remove
News - Business News	Remove
News - Business News - Fiscal Policy News	Remove
News - Newspapers	Remove
News - Politics	Remove
News - Sports News	Remove
Sports	Remove
Sports - World Sports Competitions - Olympics	Remove

Add categories Google does not associate sensitive interest categories with your ads preferences.

Opt out Opt out if you prefer ads not to be based on interests and demographics.

Opt out

Google Ads Preferences in a Fresh Instance of Safari After Browsing the New York Times

Vibrant Media

Vibrant Media is a contextual advertising network that primarily offers in-text and display advertising. We found conclusive evidence that Vibrant deliberately circumvents Safari's third-party cookie blocking feature: one of the URLs involved in the circumvention is for the resource `/safari.jsp`. The following steps describe the circumvention technology as deployed on `answers.com`. We observed identical behavior at the various region-specific subdomains of `cbslocal.com`.

1. Vibrant's main advertising script loads from http://answers.us.intellitxt.com/intellitxt/front.asp?ipid=31690. When the browser has a Safari `User-Agent` string and no Vibrant cookie, the script includes this code:

```
(function()

{try

{var e=document.createElement('iframe');e.style.display='none';e.src='ht
```

2. The Safari-specific code executes, adding an invisible `iframe` to the page.

```
<iframe style="display: none;" src="http://answers.us.intellitext.com/sa
```

The `Request-URI` parameter `t` is the current time in milliseconds, presumably used to prevent caching.

3. The `iframe` contains a `form` and a `body` `onload` handler that submits the `form`.

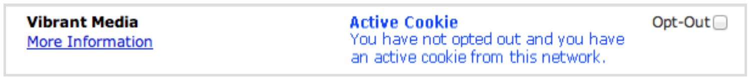
```
<html><head></head><body onload="document.getElementById('myform').submi
```

4. The response to the form contains no content and an instruction to set a Vibrant ID cookie.

```
Set-Cookie: VM_USR=AG75nlrejUwdiE6n3+naS1YAADwZAAA8VQEAAAE1gRigFAA-; Dom
```

The `Request-URI` parameter `x=1` appears to control whether the response includes the form page or a `Set-Cookie` header.

We confirmed that the "VM_USR" cookie is a Vibrant ID by checking the Network Advertising Initiative's cookie status page.


Vibrant Media
More Information
Active Cookie
You have not opted out and you have an active cookie from this network.
Opt-Out ☐

Active Vibrant Media Cookie Status Indicator

We verified that the NAI indicator is based on the presence of a valid "VM_USR" cookie, not the presence of any cookie or any "VM_USR" cookie.

Media Innovation Group

Media Innovation Group (MIG) is an advertising technology provider within the WPP family of companies. MIG's "Zeus Advertising Platform" (ZAP) is WPP's "integrated advertising and analytics platform". According to a report from a vendor, ZAP "is one of the cornerstone products created by MIG" that "provides a holistic view of site analytics and campaign data for a comprehensive understanding of every individual consumer." ZAP "collects and stores over 13 months of historical user-level data and draws from it to provide complex and robust analysis." With ZAP, "MIG is currently tracking the effectiveness of every single advertising element within many live campaigns that reach hundreds of millions of unique users per month . . . ."

We found that some MIG advertising content included a script that circumvents Safari's cookie blocking feature. Here is the relevant part of one such script we discovered at http://b3.mookie1.com/2/B3DM/DLX/1672705484@x71. A few clarifying notes: `mookie1.com` is a MIG domain (go figure), `is_http` stores whether the content is loaded over HTTP, and `ZAP_id` stores the "id" cookie.

```
if(is_http) {

    if(ZAP_id.indexOf(':') != -1 || ZAP_id == '') {

        var firstTimeSession = 0;



        function submitSessionForm() {

            if (firstTimeSession == 0) {

                firstTimeSession = 1;

                $("#sessionform").submit();

                //setTimeout(processApplication(),2000);
            }

        }



        $("body").append('<iframe id="sessionframe" name="sessionframe" onloa


        function processApplication() {

        }

    }
```

The script creates an invisible `iframe` and `form`, then submits the `form` into the `iframe` during the `onload` handler for the `iframe`.

In response to the form submission, MIG sets cookies and redirects to a 1×1 GIF.

```
$ curl -i -L -X POST "http://t.mookie1.com/t/v1/imp?"

HTTP/1.1 302 Found

Date: Fri, 17 Feb 2012 09:48:03 GMT

Server: Apache/2.0.52 (Red Hat)

Cache-Control: no-cache

Pragma: no-cache

P3P: CP="NOI DSP COR NID CUR OUR NOR"

Set-Cookie: id=3025894295853070; path=/; expires=Mon, 18-Mar-13 09:48:03 GMT;

Set-Cookie: mdata=1|3025894295853070|1329472083; path=/; expires=Mon, 18-Mar-

Set-Cookie: OAX=nVuS508+IlMACEDl; path=/; expires=Mon, 18-Mar-13 09:48:03 GMT

Location: /t/v1/imp/cc?

Content-Length: 277

Connection: close

Content-Type: text/html; charset=iso-8859-1



HTTP/1.1 200 OK

Date: Fri, 17 Feb 2012 09:48:03 GMT

Server: Apache/2.0.52 (Red Hat)

Cache-Control: no-cache

Pragma: no-cache

P3P: CP="NOI DSP COR NID CUR OUR NOR"
```

```
Set-Cookie: id=914844815541839; path=/; expires=Mon, 18-Mar-13 09:48:03 GMT;

Set-Cookie: mdata=1|914844815541839|1329472083; path=/; expires=Mon, 18-Mar-1

Set-Cookie: OAX=T6AK5U8+IlMACoIB; path=/; expires=Mon, 18-Mar-13 09:48:03 GM

Content-Length: 35

Connection: close

Content-Type: image/gif



GIF87a???????,D;
```

Comments in MIG’s script indicate that “id” is the ZAP ID cookie and “OAX” is the ID cookie for WPP’s B3 advertising optimization and custom marketplace product. We verified that the script sets a tracking cookie with MIG’s NAI status indicator.

MIG’s circumvention code appeared (relatively) infrequently; our crawl of the Alexa U.S. top 500 homepages located it on five sites. It is unclear whether MIG served this script only to Safari users. While we did not see the MIG code in any non-Safari browsers, that may have been due to insufficient sample size; we were not able to reliably cause the MIG code to appear.

That said, we believe it is nevertheless reasonable to infer that MIG’s circumvention was intentional. MIG’s code appears to be based on widely-cited sample code by web developer Anant Garg. Even Facebook’s developer documentation points to the sample.

```
var isSafari = (/Safari/.test(navigator.userAgent));

var firstTimeSession = 0;



function submitSessionForm() {

        if (firstTimeSession == 0) {

                firstTimeSession = 1;

                $("#sessionform").submit();

                setTimeout(processApplication(),2000);

        }

}



if (isSafari) {

        $("body").append('<iframe id="sessionframe" name="sessionframe" onloa

} else {

        processApplication();

}
```

```
function processApplication() {

        alert('Session has been set. Now you can start your application!');

}
```

The resemblance is uncanny. The scripts share the *exact same* variable names, structure, logic, and library dependency (jQuery 1.3.2 on Google Libraries). Even more compelling, MIG commented out a line that it didn't need!

PointRoll

PointRoll is a rich media advertising company owned by Gannett. PointRoll's corporate website claims that it "[p]ower[s] 55% of all rich media campaigns online" and "serv[es] over 450 billion impressions for more than two-thirds of the Fortune 500 brands . . . ."

We found that a PointRoll cookie helper script circumvents Safari's cookie blocking. One instance of the script we studied is at http://ads.pointroll.com/PortalServe/?pid=1574300Y14520120126002933&flash=11&time=4|13:53|-8&redir=http://at.atwola.com/adlink/5113/2209587/0/2392/AdId=232

Here's the relevant part of the script. Unlike the other examples, this code was passed through a formatter—it's otherwise unreadable.

```
function submitSessionForm(name) {

    var sessionForm = document.getElementById(name);

    if (typeof (sessionForm) != 'undefined') {

        var txtStatus = document.getElementById('txt_' + name);

        if (txtStatus.value == 'UNSUBMITTED') {

            txtStatus.value = 'SUBMITTED';

            console.log("form " + name + " Submitted");

            sessionForm.submit();

        }

    }

}

function prCook(name, value, date, dom) {

    console.log("add form: name=" + name + ": value=" + value + ": date=" + d

    var date = (typeof (date) != "undefined") ? date : "Fri, 14-Feb-2014 14:4

    var dom = (typeof (dom) != "undefined") ? dom : "ads.pointroll.com";

    var sCook = '<iframe id="' + name + '_frame" name="' + name + '_frame" or

    sCook += name;

    sCook += '')" src="http://ads.pointroll.com/clients/pointroll/cookie/blar

    sCook += '<form id="';

    sCook += name;
```

```
        sCook += ' style="display:none;" enctype="application/x-www-form-urlenco

        sCook += '<input type="text" name="name" value="' + name + '" />';

        sCook += '<input type="text" name="date" value="' + date + '" />';

        sCook += '<input type="text" name="value" value="' + value + '" />';

        sCook += '<input type="text" name="domain" value="' + dom + '" />';

        sCook += '<input type="text" id="txt_';

        sCook += name;

        sCook += '" status" value="UNSUBMITTED" />';

        sCook += '</form>';

        var d = document.createElement('DIV'),

            p = document.getElementsByTagName('BODY');

        d.innerHTML = sCook;

        if (p.length < 1) {

            p = document.getElementsByTagName('HTML');

        }

        p[0].appendChild(d);
```

The script provides a cookie setting function, `prCook`. When called, the function creates a new `div` and places within it an invisible `iframe` and `form` with the cookie fields specified by the input parameters. An `onload` handler on the `iframe` submits the `form` into the `iframe`. For example, the call

```
prCook('example_cookie_name', 'example_cookie_value', 'Fri, 14-Feb-2014 14:4
```

would result in this code being added in a new `div` element:

```
<iframe id="example_cookie_name_frame" name="example_cookie_name_frame" onloa

<form id="example_cookie_name" style="display:none" enctype="application/x-ww

   <input type="text" name="name" value="example_cookie_name" />

   <input type="text" name="date" value="Fri, 14-Feb-2014 14:47:07 GMT" />

   <input type="text" name="value" value="example_cookie_value" />

   <input type="text" name="domain" value="exampledomain.com" />

   <input type="text" id="txt_example_cookie_name" status" value="UNSUBMITTE

</form>
```

Here is the response when the `form` is submitted.

```
$ curl -i "http://ads.pointroll.com/clients/pointroll/cookie/drop.ashx?name=e

HTTP/1.1 200 OK

Connection: close

Date: Fri, 17 Feb 2012 07:41:13 GMT

Server: Microsoft-IIS/6.0

P3P: CP="NOI DSP COR PSAo PSDo OUR BUS OTC"

Access-Control-Allow-Origin: *

X-AspNet-Version: 2.0.50727

Pragma: no-cache

p3p: CP="IDC DSP COR ADM DEVi TAIi PSA PSD IVAi IVDi CONi HIS OUR IND CNT"

Set-Cookie: example_cookie_name=example_cookie_value; domain=exampledomain.co

Cache-Control: private

Content-Type: text/html; charset=utf-8

Content-Length: 145



<script>console.log('drop details: cookie name=example_cookie_name; cookie va
```

The response includes a `Set-Cookie` header with the values from the form. (And introduces a cross-site scripting vulnerability.)

PointRoll's script includes code for setting 9 cookies.

```
prCook('PRID', 'EC7B3EB9-DE19-4A32-AC07-83856AB7AE98', 'Fri, 14-Feb-2014 14:4

prCook('PRbu', 'EuQHJKn7z', 'Fri, 14-Feb-2014 14:47:07 GMT', '.pointroll.com'

prCook('PRgo', 'BAA', 'Fri, 14-Feb-2014 14:47:07 GMT', '.pointroll.com');

prCook('PRimp', '01B90400-2AC9-F37E-0209-DAC000190101', 'Fri, 14-Feb-2014 14:

prCook('PRca', '|AK5C*37611:1|#', 'Fri, 14-Feb-2014 14:47:07 GMT', 'ads.point

prCook('PRcp', '|AK5CAJmd:1|#', 'Fri, 14-Feb-2014 14:47:07 GMT', 'ads.pointro

prCook('PRpl', '|Fhtv:1|#', 'Fri, 14-Feb-2014 14:47:07 GMT', 'ads.pointroll.c

prCook('PRcr', '|GTnd:1|#', 'Fri, 14-Feb-2014 14:47:07 GMT', 'ads.pointroll.c

prCook('PRpc', '|FhtvGTnd:1|#', 'Fri, 14-Feb-2014 14:47:07 GMT', 'ads.pointro
```

It would seem reasonable to infer that the two-year "PRID" cookie is PointRoll's unique ID cookie.

As with the MIG code, we did not have a large enough sample size to conclusively determine that this script was sent only to Safari browsers. But, again, we were able to deduce that the cookie blocking circumvention was intentional by comparison to Anant Garg's example code. There are several telltale signs of copying.

- **Structure** The script is structured in the same way: an `onload` handler function followed by the `iframe` and `form`.
- **Variable Names** Both use the variable `sessionForm` and the handler function `submitSessionForm`.
- **HTML Elements** Both use the same attributes, attribute ordering, and coding style in their `iframe` and `form` elements. (The PointRoll code includes an additional `style` attribute in its `form` element.)
- **POST Bug** If you're still unconvinced, here's the giveaway: the scripts have the same bug. Both say `action="post"` (which does nothing, apparently) instead of `method="post"` (which sets the form to use the POST method instead of the GET method). The author of the example code corrected himself in a comment.

> "action" is correct for the target script in an HTML form. However, there is a second "action" in the code that should be "method" instead: method="post" (not action="post").

We found PointRoll's code on 19 homepages in the Alexa U.S. top 500.

Conclusion

When Apple's developers implemented Safari's cookie blocking feature, they were balancing several conflicting design priorities. But one decision was clear: it should prevent advertising companies from tracking the user. As a lead developer noted, whatever the implications for other websites, "I do think it would typically stop, say, doubleclick.net from tracking you . . . ."

Four advertising companies circumvented Apple's protection. Some privacy researchers and advocates have characterized the interplay between third-party web trackers and browser privacy measures as a "cat and mouse game" or "arms race." This research result regrettably affirms that view as reality—for, quite possibly, millions of users.

---

1. This post does not address the merits of Safari's cookie blocking policy.

2. There are some options for privacy-preserving social personalization. But they are far beyond the scope of this post.